L2: Entry 5 of 14 File: USPT Aug 18, 1998

DOCUMENT-IDENTIFIER: US 5796996 A
TITLE: Processor apparatus and its control method for controlling a processor having a CPU for executing an instruction according to a control program

Abstract Text (1):
In the case where a CPU executes a write instruction of a control program for a memory mapped register of an external memory, a write address and write data are written into an output buffer, thereby completing the write instruction. Prior to executing a read instruction subsequent to the write instruction, the write address and the write data of the output buffer are transferred to a sync buffer and are stored into a write address holding register and a write data holding register. Further, a using state display register is set into the holding state. When the CPU executes the read instruction, the write data of the write data holding register is written into the memory mapped register and the end of the writing operation is synchronized with the end of the read instruction. When the sync buffer unit receives an interruption instruction in the holding state, an interruption return instruction address is returned to an address of the write instruction of the data in the holding state. The process is restarted from the write instruction of the memory mapped register by the control program by the end of the interruption.

Brief Summary Text (6):
The problem regarding the recovery process of the write abnormality of the memory mapped register 750 can be solved by synchronizing the completion of the writing operation to the memory mapped register 750 with the completion of the execution of the CPU instruction instead of the conventional method whereby they are asynchronously performed. Specifically speaking, it is sufficient to provide an access instruction which doesn't pass through the output buffer and to execute the recovery process. However, although such an access method is effective for the internal output buffer 710 which is provided in the CPU 700 and can be directly accessed, it is not generally effective to the external output buffer 730 such as a secondary cache unit 720 or the like provided out of the CPU 700. Therefore, an access method in which a forced ejecting function is given to the output buffer is used. According to the access method in which the forced ejecting function is given to the output buffer, the operating state is set to a state in which an external interruption to the memory mapped register 750 is not accepted by the CPU 700, namely, into an interruption masking state, and the writing operation from the output buffer to the memory mapped register 750 is executed by the following procedure.

Brief Summary Text (9):
III. The writing operation of the memory mapped register is executed synchronously with the execution of the CPU instruction by the forced ejection of the output buffer.

Brief Summary Text (12):
FIG. 4 shows the writing operation of the memory mapped register by the forced ejecting function of the output buffer. First, the CPU 700 performs a write instruction execution 821 of the memory mapped register 750 in the interruption masking state, stores the write address and write data into the internal output buffer 710, and finishes the instruction execution. Subsequently, the CPU 700 executes the read instruction of the memory mapped register 750. Prior to executing the read instruction, a transfer writing operation 822 from the internal output buffer 710 to the external output buffer 730 is executed. For example, the transfer writing operation 822 is executed at an instruction fetching cycle for fetching the next read instruction in the CPU 700. Subsequently, when the CPU 700 executes the read instruction, a bus operation by the execution of the read instruction is detected by the secondary cache unit 720 and a writing operation 823 to an access target to write the data held in the external output buffer 730 to the memory mapped register 750 is executed. When the writing

operation 823 of the memory mapped register 750 is normally finished, the completion of the ejection is notified to the CPU 700 together with a response of the read data by the execution of the read instruction. On the other hand, when there is an abnormality in the writing operation of the memory mapped register 750, the write abnormality is notified to the CPU 700 synchronously with the response of the read data by the execution of the read instruction. The CPU 700 recognizes the write abnormality to the memory mapped register 750 in a real-time manner and executes the rewriting operation of the abnormality recovery. Therefore, when the write abnormality is notified during the writing operation corresponding to a plurality of write instructions, the subsequent instruction sequence is changed and it is possible to certain write into the memory mapped register 750.

Brief Summary Text (17):
When an external interruption for the memory mapped register occurs, in the case where the data holding state is discriminated by referring to the using state display register of the sync buffer circuit section, an interruption program processing section of the CPU returns a return instruction address to the control program at the end of the interruption to an address of the write instruction of the write address and write data in the holding state and, after completion of the interrupting process, the control program is executed from the write instruction address which was returned. Specifically speaking, after the write instruction of the memory mapped register had been finished by the control program processing section, in the case where an external interruption occurs for a period of time until the read instruction is executed after the write data and write address of the output buffer unit were transferred to the sync buffer circuit section before the next read instruction, the interruption program processing section reads out the using state display register of the sync buffer circuit section. When the sync buffer circuit section is in a holding state, the instruction address of the interruption recovery is returned by one instruction and the interrupting process for the memory mapped address is executed, thereby restarting from the write instruction of the memory mapped register by the end of the interruption.

Detailed Description Text (5):
Referring again to FIG. 6, a secondary cache memory 26-1 is provided in the outside of the CPU 22-1. The secondary cache memory 26-1 is controlled by a secondary cache controller 28. The CPU 22-2 and secondary cache memory 26-2 can be extended as necessary for the CPUs 22-1 and 26-1. Subsequent to the secondary cache controller 28, a local storage unit 34 is provided through a memory access unit 30. The local storage unit 34 becomes an external memory for the CPU 22-1. Further, an ROM 36 in which various kinds of control programs to be executed by the CPU 22-1 have been stored is provided. A memory mapped register 68 is provided for the local storage unit 34 in correspondence to the CPU 22-1. The memory mapped register 68 is used as a control register of the I/O control apparatuses 18-1 to 18-4 in FIG. 5. The CPU 22-1 executes a memory mapped register write instruction of the control program, thereby writing data into the memory mapped register 68 of the local storage unit 34. Generally, the write instruction for the memory mapped register 68 is performed by sequentially executing a plurality of write instructions. When the write instruction for the memory mapped register 68 as a target is executed a plurality of number of times and the writing operation is completed, the CPU 22-1 executes a read instruction of the memory mapped register 68 and executes the control operation for the I/O control apparatus on the basis of the read data. The execution of the write instruction of the memory mapped register 68 in the CPU 22-1 is finished by storing a write address and write data into the storage buffer 24-1 which functions as an internal output buffer. Prior to executing the read instruction for the memory mapped register 68 by the next CPU 22-1, the write address and write data stored in the storage buffer 24-1 are transferred to the secondary cache memory 26-1 and, further, transferred to a sync buffer circuit section 32 in the memory access unit 30 newly provided in the invention.

Detailed Description Text (10):
When the control program processing section 62 executes, for example, a write instruction of the memory mapped register 68 of the local storage unit 34, the write address and write data are stored into a storage buffer 24 serving as an internal output buffer, so that the write instruction is finished. Ordinarily, the writing to the memory mapped register 68 is completed by executing a plurality of write instructions. Therefore, the write addresses and write data of the number corresponding to the number of write instructions are stored into the storage buffer 24. After completion of the execution of the plurality of write instructions, the control program processing section 62 executes the read instruction of the memory mapped register 68. Prior to executing the read instruction, the write addresses and write data stored in the storage buffer 24 are transferred to the sync buffer circuit section 32 provided

for the memory access unit 30 via an external output buffer 26 which is realized by the cache memory of the secondary cache controller 28. The write address and write data transferred from the storage buffer 24 are stored into the write address holding register 56 and write data holding register 58 by the register control circuit 50. In this instance, the busy bit 55 of the using state display register 54 is set to bit 1, thereby indicating that the sync buffer circuit section 32 is in a using state. The data transfer from the storage buffer 24 is executed at a timing when a fetching cycle operation of the read instruction to be executed next in the control program processing section 62 is detected. Subsequently, the read instruction advances to the executing cycle and the read instruction is executed. The execution of the read instruction is realized by transferring a read command and a read address to the memory access unit 30 through an internal bus. In response to the execution of the read instruction by the CPU 22, in the sync buffer circuit section 32, the bus operation of the internal bus in association with the read instruction is detected by the CPU instruction detecting and decoding circuit 48 (refer to FIG. 8). On the basis of the detection of the bus operation, the register control circuit 50 reads out the write address in the write address holding register 56 and the write data in the write data holding register 58 and instructs a writing operation to the local storage unit 34. Thus, the writing operation of the write data transferred to a memory mapped register 68-1 designated by the write address is executed. At the same time, since a read command of the memory mapped register 68-1 has been sent from the CPU 22 to the memory access controller 30, read data has to be responded. In the ordinary reading operation, the data is read out from the memory mapped register 68-1 of the local storage unit 34. However, the data written in the memory mapped register 68-1 in this instance is the write data of the write data holding register 58 of the sync buffer circuit section. Therefore, the reading operation from the memory mapped register 68-1 is not executed but after completion of the writing operation of the write data of the write data holding register 58, the write data in the write data holding register 58 is read out and is responded to the CPU as read data of a dummy via the internal bus.

Detailed Description Text (19):
Processes of the interruption program processing section 64 in the case where an external interruption occurred for the same memory mapped register 68 during the execution of the write instruction and read instruction for the memory mapped register 68 by the control program processing section 62 provided for the CPU 22 will now be described. FIG. 18 is a time chart in the case where the control program processing section 62 executes another instructing operation in block 500 and, before the write instructing operation for the memory mapped register is subsequently executed in block 502, an external interruption in block 501 occurs. When the external interruption in block 501 occurs before the write instructing operation for the memory mapped register in block 502 as mentioned above, the interruption program processing section 64 first reads out the using state display register (status register) 54 of the sync buffer circuit section 32 in block 600. In this instance, since the write instructing operation for the memory mapped register in block 502 and the read instructing operation in block 503 are not executed, the sync buffer circuit section 32 is in a data non-holding state as shown in block 601. Namely, when checking the busy bit 55 due to the reading of the using state display register 54, the busy bit 55 is set to bit 0. In this case, in block 602, an interruption return instruction address is maintained as an instruction address of the write instructing operation in next block 502 of the occurrence of the external interruption in block 501 as it is. An interrupting process is executed in block 603. As for the interrupting process in block 603, in a manner similar to the process by the control program processing section 62, as shown in FIG. 12, the interruption program processing section 64 of the CPU 22 executes the write instruction of the memory mapped register 68 and stores the write data and read data into the storage buffer 24. Prior to the next read instruction of the memory mapped register 68, as shown in FIG. 13, after the write address and write data were transferred to the sync buffer circuit section 32 through the secondary cache memory (external output buffer) 26 from the storage buffer 24 and stored into the write address holding register 56 and write data holding register 58, the busy bit 55 of the using state display register 54 is turned on to bit 1. Subsequently, as shown in FIG. 14, when the read instruction of the memory mapped register 68 is executed in the CPU 22 by the control program processing section 62, the reading operation of the bus in association with it is detected by the sync buffer circuit section 32. The holding address and read address are compared. Since the coincidence of those addresses is obtained, the writing operation of the holding data for the memory mapped register 68 is executed. After completion of the interrupting process for the memory mapped register 68 in block 603 as mentioned above, the address is returned in block 604 to the interruption return instruction address set in block 602. After completion of the interruption, the control program processing section 62 executes the instructing

operation for the memory mapped register in block 502.

WEST

Generate Collection Print

L2: Entry 7 of 14 File: USPT Oct 13, 1992

DOCUMENT-IDENTIFIER: US 5155832 A
TITLE: Method to increase performance in a multi-level cache system by the use of forced cache misses

Abstract Text (1):
A computing system includes a processor, a system memory containing data utilized by the processor and two cache memories. Each cache memory is connected directly to the processor. A first cache memory is connected to the processor. A second cache memory is connected to the processor and to the system memory. The second cache memory contains a subset of data in the system memory. The first cache memory contains a subset of data in the second cache memory. Data integrity in the system memory is maintained using the second cache memory only. During the execution of a first instruction data required for execution of the first instruction might not be available in the first cache memory. The data required for execution of the first instruction is obtained from the second cache memory and written into the first cache memory. If, however, there is an attempt to access from the first cache memory data required for a second instruction, and this attempt to access the first cache memory occurs simultaneously to the time when the data required for execution of the first instruction is being written from the school cache memory to the first cache memory, then a cache miss is forced and the second cache memory is accessed for the data required for execution of the second instruction.

Brief Summary Text (10):
When data is executed in pipelined stages, a conflict may occur when different pipeline stages for two separate instructions both call for access of the first cache memory. For example, during the execution of a first instruction data required for execution of the first instruction might not be available in the first cache memory. As a result of this miss in the first cache memory, the data required for execution of the first instruction is obtained from the second cache memory and written into the first cache memory. If, however, there is an attempt to access from the first cache memory data required for a second instruction, and this attempt to access the first cache memory occurs simultaneously to the time when the data required for execution of the first instruction is being written from the second cache memory to the first cache memory, a conflict will occur. Rather than delaying execution of the second instruction, until the first cache memory is free, a cache miss may be forced. That is, execution proceeds as though the first cache memory does not have the data required for execution of the second instruction, and the second cache memory is accessed for the data required for execution of the second instruction.

Detailed Description Text (13):
As discussed above, CPU 101 accesses cache memory 102 and cache memory 103 simultaneously. In order to additionally increase execution efficiency, CPU 101 may continue executing instructions past a load operation which has not completed because the data sought was not in cache memory 102. This may be done unless CPU 101 encounters an instruction which requires the result of the load before the load completes.

Detailed Description Text (18):
When there is a cache memory miss for data being fetched from cache memory 102 and there is no immediate need for the data, CPU 101 may continue executing instructions past the operation which has not completed. In order to illustrate this, cache memory access stages Cache Stage I and Cache Stage II are shown in Table 3. Stages Cache Stage I and Cache Stage II represent the cycles necessary to retrieve data from cache memory 103 when there is a miss in cache memory 102. Cache Stage I and Cache Stage II are pseudo stages which are only utilized by instructions when there is a miss at cache memory 102.

Detailed Description Text (19):
Table 3 illustrates what happens if CPU 101 is continuing execution past a load which has not yet completed. In cycle 2, Load A has a cache memory miss of data at cache memory 102. In cycle 3 the instruction "Load A" obtains data from cache memory 103. In cycle 4, "Load A" copies the data from cache memory 103 into cache memory 102. However, Load B is in the "Memory Stage" and is attempting to read data from cache memory 102. Since both instructions cannot simultaneously access cache memory 102 during cycle 4, the normal operation of the machine is halted until "Load A" has completed copying data. In cycle 5 the instruction "Load B" is able to access cache memory 102 and normal operation is resumed.

Detailed Description Text (22):
An alternate solution to that presentented in Table 4 is presented in Table 5 below. In cycle 4 of Table 5, both the instruction "Load A" and the instruction "Load B" are attempting to access cache memory 102. Instead of allowing the instruction "Load A" to complete and forcing a miss on cache memory 102 of instuction "Load B" as in the example illustrated by Table 4, the instruction "Load B" is allowed to access cache memory 102 and the instruction "Load A" is prevented from putting data into cache memory 102.

L6: Entry 5 of 7 File: USPT Jun 3, 1997

DOCUMENT-IDENTIFIER: US 5636363 A
TITLE: Hardware control structure and method for off-chip monitoring entries of an on-chip cache

Brief Summary Text (6):
In the course of developing or debugging a computer system, it is often necessary to monitor program execution by the CPU or to interrupt one instruction stream to direct the CPU to execute certain alternate instructions. For example, a technique for testing a microprocessor in a system under development uses an in-circuit emulator (ICE) which monitors the instruction stream and where appropriate, takes control of the microprocessor by forcing the microprocessor to execute an alternative program. To achieve this end, the ICE monitors the signals on the microprocessor's pins. When a monitored instruction in the program execution is encountered, the ICE causes alternate instructions to be executed for such purpose as reading or altering the internal state of the CPU. Hence, when the cache memory is implemented off-chip, the transactions between the cache memory and the CPU can be monitored by the ICE via the microprocessor's pins on the off-chip bus between the cache memory and the CPU. However, when the cache memory is implemented on-chip, the transactions between the cache and the CPU occur on an internal on-chip bus, which cannot be probed from the pins of the integrated circuit. As a result, debugging operations using an ICE in a system with an on-chip cache memory can be very difficult. When the on-chip cache memory achieves a high hit ratio, only the relatively infrequent accesses to main memory due to cache misses or references to uncacheable portions of memory can be monitored from the pins.

Brief Summary Text (10):
In a microprocessor with an on-chip cache, a structure and a method are provided to direct execution of instructions provided on a memory bus between the microprocessor and the main memory. In accordance with the present invention, when a testing device desires to provide on the memory bus an instruction to be executed by the microprocessor, a cache miss signal is generated to force the processor to fetch the next instruction from the main memory. The instruction to be executed is then provided to the microprocessor as if the instruction is fetched from the main memory.

WEST

Generate Collection Print

L10: Entry 9 of 26 File: USPT Oct 10, 2000

DOCUMENT-IDENTIFIER: US 6131155 A
TITLE: Programmer-visible uncached load/store unit having burst capability

Detailed Description Text (35):
Referring to FIG. 6, if the dirty flag is set, the Buffer Flush instruction forces the uncached load/store unit to initiate the transfer of the data in the buffer RAM to the main memory, and also prevents the CPU from executing any additional instructions of the program until the transfer is complete and the dirty flag has been cleared. If the dirty flag is clear, the instruction has no effect, as this guarantees that the main memory contains the latest copy of any uncached store data.

Current US Original Classification (1):
712/207

Current US Cross Reference Classification (1):
711/138

WEST Search History

DATE: Thursday, September 25, 2003

Set Name side by side	Query	Hit Count	Set Name result set
DB=JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L12	(force$2 with execut$6 with memory with instruction$2 with barrier$2)	0	L12
L11	(force$2 with execut$6 with memory with instruction$2) and (complet$6 with memory with instruction$2)	2	L11
DB=USPT; PLUR=YES; OP=ADJ			
L10	L9 and l1	26	L10
L9	(711/$)!.CCLS. or 712/$.ccls.	21885	L9
L8	L7 and l1	3	L8
L7	((717/$)!.CCLS.)	3810	L7
L6	L4 and (interrupt with CPU$2)	7	L6
L5	L4 and (force$2 with interrupt with CPU$2)	0	L5
L4	(force$2 with execut$6 with memory with instruction$2)	56	L4
L3	(force$2 with execut$6 with memory with instruction$2 with barrier)	0	L3
L2	(force$2 with execut$6 with memory with instruction$2) and (complet$6 with memory with instruction$2)	14	L2
L1	(force$2 with execut$6 with memory with instruction$2)	56	L1

END OF SEARCH HISTORY